

FFFFFFFFFFFFFFFF	111	111	XXX	XXX
FFFFFFFFFFFFFFFF	111	111	XXX	XXX
FFFFFFFFFFFFFFFF	111	111	XXX	XXX
FFF	111111	111111	XXX	XXX
FFF	111111	111111	XXX	XXX
FFF	111111	111111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFFFFFFF FFF	111	111	XXX	XXX
FFFFFFFFFFFF	111	111	XXX	XXX
FFFFFFFFFFFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	1111111111	1111111111	XXX	XXX
FFF	1111111111	1111111111	XXX	XXX
FFF	1111111111	1111111111	XXX	XXX

IOCS
IO-C
IO-C
IO-D
IO-F
IO-S
KICL

KILL
KILL
LB_E
LB_C
LB_F
LB_H
LB_L
LOCA
LOCA
LOCK

LOCK
LOCK
LOCK
LOC
LOC
L_CC
L_CC
L_DA
L_DA
MAIN
MAKE
MAKE
MAKE
MAKE
MAKE

MAKE
MAKE
MAP
MAP

MAP
MAR
MAR
MAR
MAR
MAR

```
BBBBBBBBB      AAAAAA      DDDDDDDD      SSSSSSSS      CCCCCCCC      NN      NN
BBBBBBBBB      AAAAAA      DDDDDDDD      SSSSSSSS      CCCCCCCC      NN      NN
BB      BB      AA      AA      DD      DD      SS      CC      NN      NN
BB      BB      AA      AA      DD      DD      SS      CC      NN      NN
BB      BB      AA      AA      DD      DD      SS      CC      NNNN      NN
BBBBBBBBB      AA      AA      DD      DD      SSSSSS      CC      NNNN      NN
BBBBBBBBB      AA      AA      DD      DD      SSSSSS      CC      NN      NN
BB      BB      AAAAAAAAAA      DD      DD      SS      CC      NN      NN
BB      BB      AAAAAAAAAA      DD      DD      SS      CC      NN      NN
BB      BB      AA      AA      DD      DD      SS      CC      NN      NN
BB      BB      AA      AA      DD      DD      SS      CC      NN      NN
BBBBBBBBB      AA      AA      DDDDDDDD      SSSSSSSS      CCCCCCCC      NN      NN
BBBBBBBBB      AA      AA      DDDDDDDD      SSSSSSSS      CCCCCCCC      NN      NN
```

```
LL      IIIIII      SSSSSSSS
LL      IIIIII      SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLL      IIIIII      SSSSSSSS
```



```
1 0001 0 MODULE BADSCN (
2 0002 0 LANGUAGE (BLISS32),
3 0003 0 IDENT = 'V04-000',
4 0004 0 ) =
5 0005 1 BEGIN
6 0006 1
7 0007 1
8 0008 1 *****
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
12 0012 1 * ALL RIGHTS RESERVED.
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
19 0019 1 * TRANSFERRED.
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
23 0023 1 * CORPORATION.
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
27 0027 1 *
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1 ++
32 0032 1
33 0033 1 FACILITY: F11ACP Structure Level 2
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1
37 0037 1 This routine scans the pending bad block list and enters or removes
38 0038 1 entries.
39 0039 1
40 0040 1 ENVIRONMENT:
41 0041 1
42 0042 1 STARLET operating system, including privileged system services
43 0043 1 and internal exec routines.
44 0044 1
45 0045 1 --
46 0046 1
47 0047 1
48 0048 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 22-May-1978 10:33
49 0049 1
50 0050 1 MODIFIED BY:
51 0051 1
52 0052 1 V03-003 CDS0001 Christian D. Saether 29-Dec-1983
53 0053 1 Use L_NORM linkage and BIND_COMMON macro.
54 0054 1
55 0055 1 V03-002 ACG0367 Andrew C. Goldstein, 26-Oct-1983 19:48
56 0056 1 Update highwater mark in BADLOG.SYS when extending
57 0057 1
```

BADSCN
V04-000

B 4
15-Sep-1984 23:55:15
14-Sep-1984 12:30:09

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]BADSCN.B32;1 Page 2 (1)

```

: 58      0058 1 | V3-001 STJ0312 Steven T. Jeffreys, 2-Jun-1982
: 59      0059 1 | Fix addressing mode to fix out-of-range reference.
: 60      0060 1 |
: 61      0061 1 | B0101 ACG0121 Andrew C. Goldstein, 16-Jan-1980 21:36
: 62      0062 1 | Make context save and restore into subroutines
: 63      0063 1 |
: 64      0064 1 | B0100 ACG00001 Andrew C. Goldstein, 10-Oct-1978 19:59
: 65      0065 1 | Previous revision history moved to [F11B.SRC]F11B.REV
: 66      0066 1 | **
: 67      0067 1 |
: 68      0068 1 |
: 69      0069 1 | LIBRARY 'SYSS$LIBRARY:LIB.L32';
: 70      0070 1 | REQUIRE 'SRC$:FCPDEF.B32';
```

CHA
V04


```
1061 1 GLOBAL ROUTINE SCAN_BADLOG (FID, BASE_VBN, BASE_LBN, MODE, BLOCK_COUNT) : L_NORM NOVALUE =
1062 1
1063 1 ++
1064 1
1065 1 FUNCTIONAL DESCRIPTION:
1066 1
1067 1 This routine scans the volume bad block log for the specified block(s)
1068 1 and enters or removes them, depending on the mode.
1069 1
1070 1
1071 1 CALLING SEQUENCE:
1072 1 SCAN_BADLOG (ARG1, ARG2, ARG3, ARG4, ARG5)
1073 1
1074 1 INPUT PARAMETERS:
1075 1 ARG1: file ID of file containing bad block
1076 1 ARG2: VBN in file of bad block
1077 1 ARG3: LBN of bad block
1078 1 ARG4: mode of operation
1079 1 REMOVE_BADBLOCK = 0
1080 1 ENTER_READERR = 1
1081 1 ENTER_WRITERR = 2
1082 1 ARG5: count of blocks to process (remove only)
1083 1
1084 1 IMPLICIT INPUTS:
1085 1 NONE
1086 1
1087 1 OUTPUT PARAMETERS:
1088 1 NONE
1089 1
1090 1 IMPLICIT OUTPUTS:
1091 1 NONE
1092 1
1093 1 ROUTINE VALUE:
1094 1 NONE
1095 1
1096 1 SIDE EFFECTS:
1097 1 volume bad block list file altered
1098 1
1099 1 --
1100 1
1101 2 BEGIN
1102 2
1103 2 MAP
1104 2 FID : REF BBLOCK; ! file ID argument
1105 2
1106 2 LABEL
1107 2 SEARCH_LOOP; ! bad block list search
1108 2
1109 2 LOCAL
1110 2 VBN, ! VBN of current block of file
1111 2 LBN, ! LBN of current block
1112 2 FIRST_FREE, ! VBN containing first free slot
1113 2 FREE_OFFSET, ! byte offset in block of free slot
1114 2 P : REF BBLOCK, ! record pointer
1115 2 FIB : REF BBLOCK, ! local pointer to secondary FIB
1116 2 WINDOW : REF BBLOCK, ! address of bad block file window
1117 2 FCB : REF BBLOCK, ! address of bad block file FCB
1118 2
```



```
129      1113 2      HEADER      : REF BBLOCK;      ! address of bad block file header
130      1119 2
131      1120 2 BIND_COMMON;
132      1121 2
133      1122 2 EXTERNAL ROUTINE
134      1123 2      SAVE_CONTEXT      : L_NORM,      ! save reentrant context area
135      1124 2      RESTORE_CONTEXT : L_NORM,      ! restore reentrant context area
136      1125 2      OPEN_FILE       : L_NORM,      ! open a data file
137      1126 2      READ_DATA        : L_NORM,      ! read a file block
138      1127 2      CLOSE_FILE       : L_NORM,      ! close the data file
139      1128 2      CREATE_BLOCK     : L_NORM,      ! fabricate a block buffer
140      1129 2      MARK_DIRTY       : L_NORM,      ! mark buffer dirty
141      1130 2      READ_HEADER      : L_NORM,      ! read file header
142      1131 2      EXTEND           : L_NORM,      ! extend a file
143      1132 2      CHECKSUM         : L_NORM,      ! checksum file header
144      1133 2      WRITE_HEADER     : L_NORM,      ! write a file header
145      1134 2      MAP_VBN         : L_NORM;      ! map virtual to logical
146      1135 2
147      1136 2
148      1137 2 ! Check the reserved file count to see if there is a bad block list file.
149      1138 2 ! If not, forget the whole thing. If there is, set up secondary context and
150      1139 2 ! open the file.
151      1140 2
152      1141 2
153      1142 2 IF .CURRENT_VCB[VCB$B_RESFILES] LSSU BADLOG_FID THEN RETURN;
154      1143 2
155      1144 2 SAVE_CONTEXT ();
156      1145 2 FIB = SECOND FIB;
157      1146 2 FIB[FIB$W_FID_NUM] = BADLOG_FID;
158      1147 2 FIB[FIB$W_FID_SEQ] = BADLOG_FID;
159      1148 2
160      1149 2 WINDOW = OPEN_FILE (FIB[FIB$W_FID], 1);
161      1150 2
162      1151 2 ! Scan the pending bad block file for a match on the given LBN (in remove mode,
163      1152 2 ! scan for the given range). Also look for the first available free space.
164      1153 2
165      1154 2
166      1155 2 FIRST_FREE = 0;
167      1156 2 VBN = 0;
168      1157 2
169      1158 2 SEARCH_LOOP: BEGIN
170      1159 3 WHILE T DO
171      1160 4 BEGIN
172      1161 4 VBN = .VBN + 1;
173      1162 4 P = READ_DATA (.WINDOW, .VBN, 1);
174      1163 4 IF .P EQ 0 THEN EXITLOOP;
175      1164 4
176      1165 4 INCR J FROM 0 TO 512/PBB$C_LENGTH - 1
177      1166 4 DO
178      1167 5 BEGIN
179      1168 5 IF .MODE EQ REMOVE_BADBLOCK
180      1169 5 THEN
181      1170 6 BEGIN
182      1171 6 IF .P[PBB$L_LBN] GEQU .BASE_LBN
183      1172 6 AND .P[PBB$L_LBN] LSSU .BASE_LBN + .BLOCK_COUNT
184      1173 6 THEN
185      1174 7 BEGIN
```

```
186      CH$FILL (0, PBB$C_LENGTH, .P);
187      MARK_DIRTY (.P);
188      END;
189      END
190      ELSE
191      BEGIN
192      IF .P[PBB$B_FLAGS] EQL 0
193      AND .FIRST_FREE EQL 0
194      THEN
195      BEGIN
196      FIRST_FREE = .VBN;
197      FREE_OFFSET = .J * PBB$C_LENGTH;
198      END;
199      IF .P[PBB$L_LBN] EQL .BASE_LBN
200      THEN
201      BEGIN
202      IF .P[PBB$B_COUNT] LSSU 255
203      THEN P[PBB$B_COUNT] = .P[PBB$B_COUNT] + 1;
204      IF .MODE EQL ENTER_READERR
205      THEN P[PBB$V_READERR] = 1;
206      ELSE P[PBB$V_WRITERR] = 1;
207      MARK_DIRTY (.P);
208      LEAVE SEARCH_LOOP;
209      END;
210      END;
211      P = .P + PBB$C_LENGTH;
212      END;
213      ! end of loop within block
214      END;
215      ! end of loop scanning blocks
216      ! We get here if we fail to match on the block (or were scanning to remove).
217      ! On a remove, we are now done. For an enter, take the first free slot found.
218      ! If there was none, we have to extend the file.
219      !
220      IF .MODE EQL REMOVE_BADBLOCK THEN LEAVE SEARCH_LOOP;
221      IF .FIRST_FREE EQL 0
222      THEN
223      BEGIN
224      FCB = .WINDOW[WCB$L_FCB];
225      HEADER = READ_HEADER (0, .FCB);
226      IF .FCB[FCB$L_EFBLK] GEQU .FCB[FCB$L_FILESIZE]
227      THEN
228      BEGIN
229      FIB[FIB$L_EXSZ] = 1;
230      FIB[FIB$V_NOHDREXT] = 1;
231      EXTEND (.FIB, .HEADER);
232      END;
233      BBLOCK [HEADER[FH2$W_RECATTR], FAT$W_EFBLKL] =
234      .BBLOCK [HEADER[FH2$W_RECATTR], FAT$W_EFBLKL] + 1;
235      ! If this file header supports it, stuff the high water field to
236      ! be the allocated size.
237      !
```



```

: 243      1232  4
: 244      1233  4      IF .HEADER [FH2$B_IDOFFSET] GEQU ($BYTEOFFSET (FH2$L_HIGHWATER)+4)/2
: 245      1234  4      THEN
: 246      1235  4          HEADER[FH2$L_HIGHWATER] = .BBLOCK [HEADER[FH2$W_RECATTR], FAT$W_EFBLKL];
: 247      1236  4
: 248      1237  4      CHECKSUM (.HEADER);
: 249      1238  4      WRITE_HEADER ();
: 250      1239  4
: 251      1240  4      LBN = MAP_VBN (.VBN, .WINDOW);
: 252      1241  4      IF .LBN EQL -1
: 253      1242  4      THEN BUG_CHECK (HDRNOTMAP, FATAL, 'Block just created not mapped');
: 254      1243  4      P = CREATE_BLOCK (.LBN, 1, DATA_TYPE);
: 255      1244  4      FREE_OFFSET = 0;
: 256      1245  4      END
: 257      1246  4
: 258      1247  3      ELSE
: 259      1248  3          P = READ_DATA (.WINDOW, .FIRST_FREE, 1);
: 260      1249  3
: 261      1250  3      ! Now build the new bad block list entry.
: 262      1251  3      !
: 263      1252  3
: 264      1253  3      P = .P + .FREE_OFFSET;
: 265      1254  3
: 266      1255  3      CH$COPY (FID$C_LENGTH, .FID, 0, PBB$C_LENGTH, P[PBB$W_FID]);
: 267      1256  3      P[PBB$L_VBN] = .BASE_VBN;
: 268      1257  3      P[PBB$L_LBN] = .BASE_LBN;
: 269      1258  3      P[PBB$B_COUNT] = .P[PBB$B_COUNT] + 1;
: 270      1259  3      IF .MODE EQL ENTER_READERR
: 271      1260  3      THEN P[PBB$V_READERR] = 1
: 272      1261  3      ELSE P[PBB$V_WRITERR] = 1;
: 273      1262  3
: 274      1263  3      MARK_DIRTY (.P);
: 275      1264  3
: 276      1265  2      END;
: 277      1266  2
: 278      1267  2      CLOSE_FILE (.WINDOW);
: 279      1268  2      RESTORE_CONTEXT ();
: 280      1269  2
: 281      1270  1      END;

```

! end of block SEARCH_LOOP

! close the bad block list file

! end of routine SCAN_BADLOG

```

.TITLE  BADSCN
.IDENT  \V04-000\

```

```

.EXTRN  SAVE_CONTEXT, RESTORE_CONTEXT
.EXTRN  OPEN_FILE, READ_DATA
.EXTRN  CLOSE_FILE, CREATE_BLOCK
.EXTRN  MARK_DIRTY, READ_HEADER
.EXTRN  EXTEND, CHECKSUM
.EXTRN  WRITE_HEADER, MAP_VBN
.EXTRN  BUG$_HDRNOTMAP

```

```

.PSECT  $CODE$,NOWRT,2

```

```

.ENTRY  SCAN_BADLOG, Save R2,R3,R4,R5,R6,R7,R8,R9,- ; 1061
        R11
        #8, SP

```

OBFC 00000

5E

08 C2 00002

	50	98	AA	D0	00005	MOVL	-104(BASE), R0	1142
	09	4F	A0	91	00009	CMPB	79(R0), #9	
			01	1E	0000D	BGEQU	1\$	
			04	00	0000F	RET		
0000G	CF		00	FB	00010	CALLS	#0, SAVE_CONTEXT	1144
	57	0244	CA	9E	00015	MOVAB	580(BASE), FIB	1145
04	A7	00090009	8F	D0	0001A	MOVL	#589833, 4(FIB)	1146
			01	DD	00022	PUSHL	#1	1149
			A7	9F	00024	PUSHAB	4(FIB)	
0000G	CF		02	FB	00027	CALLS	#2, OPEN_FILE	
	5B		50	D0	0002C	MOVL	R0, WINDOW	
			59	D4	0002F	CLRL	FIRST_FREE	1155
			6E	D4	00031	CLRL	VBN	1156
			6E	D6	00033	INCL	VBN	1161
			01	DD	00035	PUSHL	#1	1162
		04	AE	DD	00037	PUSHL	VBN	
			5B	DD	0003A	PUSHL	WINDOW	
0000G	CF		03	FB	0003C	CALLS	#3, READ_DATA	
	56		50	D0	00041	MOVL	R0, P	
			57	13	00044	BEQL	8\$	1163
			58	D4	00046	CLRL	J	1165
		10	AC	D5	00048	TSTL	MODE	1168
			22	12	0004B	BNEQ	4\$	
	OC	AC	OC	A6	D1	CMPL	12(P), BASE_LBN	1171
			40	1F	00052	BLSSU	7\$	
50	OC	AC	14	AC	C1	ADDL3	BLOCK_COUNT, BASE_LBN, R0	1172
	50		OC	A6	D1	CMPL	12(P), R0	
			34	1E	0005E	BGEQU	7\$	
10	00	6E	00	2C	00060	MOVCS	#0, (SP), #0, #16, (P)	1175
			66		00065			
			56	DD	00066	PUSHL	P	1176
0000G	CF		01	FB	00068	CALLS	#1, MARK_DIRTY	
			25	11	0006D	BRB	7\$	1168
		06	A6	95	0006F	TSTB	6(P)	1181
			OC	12	00072	BNEQ	5\$	
			59	D5	00074	TSTL	FIRST_FREE	1182
			08	12	00076	BNEQ	5\$	
	59		6E	D0	00078	MOVL	VBN, FIRST_FREE	1185
04	AE		04	78	0007B	ASHL	#4, J, FREE_OFFSET	1186
			OC	A6	D1	CMPL	12(P), BASE_LBN	1188
			0D	12	00085	BNEQ	7\$	
	FF	8F	07	A6	91	CMPB	7(P), #255	1191
			03	1F	0008C	BLSSU	6\$	
			00A4	31	0008E	BRW	16\$	
			009E	31	00091	BRW	15\$	
	56		10	C0	00094	ADDL2	#16, P	1201
AD	58		1F	F3	00097	AOBLEQ	#31, J, 3\$	1165
			96	11	0009B	BRB	2\$	1159
		10	AC	D5	0009D	TSTL	MODE	1210
			03	12	000A0	BNEQ	9\$	
			00A7	31	000A2	BRW	19\$	
			59	D5	000A5	TSTL	FIRST_FREE	1212
			6B	12	000A7	BNEQ	13\$	
	53	18	AB	D0	000A9	MOVL	24(WINDOW), FCB	1215
			53	DD	000AD	PUSHL	FCB	1216
			7E	D4	000AF	CLRL	-(SP)	
0000G	CF		02	FB	000B1	CALLS	#2, READ_HEADER	

38	52	3C	50	D0	000B6	MOVL	R0, HEADER	1218
A3	A3	A3	D1	000B9	CMPL	60(FCB), 56(FCB)		
18	A7	01	1F	000BE	BLSSU	10\$		
17	A7	02	D0	000C0	MOVL	#1, 24(FIB)		1221
		02	88	000C4	BISB2	#2, 23(FIB)		1222
		52	DD	000C8	PUSHL	HEADER		1223
		57	DD	000CA	PUSHL	FIB		
0000G	CF	02	FB	000CC	CALLS	#2, EXTEND		
	28	1E	A2	B6	INCW	30(HEADER)		1227
		62	91	000D4	CMPB	(HEADER), #40		1233
		05	1F	000D7	BLSSU	11\$		
4C	A2	1E	A2	3C	MOVZWL	30(HEADER), 76(HEADER)		1235
		52	DD	000DE	PUSHL	HEADER		1237
0000G	CF	01	FB	000E0	CALLS	#1, CHECKSUM		
0000G	CF	00	FB	000E5	CALLS	#0, WRITE_HEADER		1238
		5B	DD	000EA	PUSHL	WINDOW		1240
		04	AE	DD	000EC	PUSHL	VCN	
0000G	CF	02	FB	000EF	CALLS	#2, MAP_VCN		
FFFFFFFF	8F	50	D1	000F4	CMPL	LBN, #-T		1241
		04	12	000FB	BNEQ	12\$		
			FEFF	000FD	BUGW			1242
			0000*	000FF	.WORD	<BUG\$_HDRNOTMAP!4>		
		04	DD	00101	PUSHL	#4		1243
		01	DD	00103	PUSHL	#1		
		50	DD	00105	PUSHL	LBN		
0000G	CF	03	FB	00107	CALLS	#3, CREATE_BLOCK		
56	56	50	D0	0010C	MOVL	R0, P		
		04	AE	D4	CLRL	FREE_OFFSET		1244
		0E	11	00112	BRB	14\$		1212
		01	DD	00114	PUSHL	#1		1248
		59	DD	00116	PUSHL	FIRST_FREE		
		5B	DD	00118	PUSHL	WINDOW		
0000G	CF	03	FB	0011A	CALLS	#3, READ_DATA		
56	56	50	D0	0011F	MOVL	R0, P		
		04	AE	C0	ADDL2	FREE_OFFSET, P		1253
10	00	04	BC	06	MOVC5	#6, BFID, #0, #16, (P)		1255
		08	A6	66				
		07	AC	7D	MOVQ	BASE_VCN, 8(P)		1256
		10	A6	96	INCB	7(P)		1258
			AC	D1	CMPL	MODE, #1		1259
		06	12	00139	BNEQ	17\$		
06	A6	01	88	0013B	BISB2	#1, 6(P)		1260
		04	11	0013F	BRB	18\$		
06	A6	02	88	00141	BISB2	#2, 6(P)		1261
		56	DD	00145	PUSHL	P		1263
0000G	CF	01	FB	00147	CALLS	#1, MARK_DIRTY		
		5B	DD	0014C	PUSHL	WINDOW		1267
0000G	CF	01	FB	0014E	CALLS	#1, CLOSE_FILE		
0000G	CF	00	FB	00153	CALLS	#0, RESTORE_CONTEXT		1268
		04	00158	RET				1270

; Routine Size: 345 bytes, Routine Base: \$CODE\$ + 0000

: 282 1271 1
: 283 1272 1 END
: 284 1273 0 ELUDOM

PSECT SUMMARY

```
:
:      Name                Bytes                Attributes
:
: $CODE$                  345 NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(2)
```

Library Statistics

```
:
:      File                Total  Symbols  Percent  Pages  Processing
:                        Total  Loaded  Percent  Mapped  Time
:
: _$255$DUA28:[SYSLIB]LIB.L32;1  18619      39      0     1000     00:02.0
```

COMMAND QUALIFIERS

```
:
: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:BADSCN/OBJ=OBJ$:BADSCN MSRC$:BADSCN/UPDATE=(ENH$:BADSCN)
```

```
: Size:          345 code + 0 data bytes
: Run Time:      00:21.3
: Elapsed Time:  00:49.9
: Lines/CPU Min: 3589
: Lexemes/CPU-Min: 43192
: Memory Used:  278 pages
: Compilation Complete
```


0168

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY